100th Anniversary of Chula Engineering 2013

Basic I/O, Data types, Variables, Operators, Expressions, Statements

python powered

Topics

- Program structures
- `print(…)`
- Variables
- Data types:
  - `int, float, bool, str, list`
- Type conversions:
  - `int(), float(), str()`
- `input()`
- Arithmetic operations:
  - `+ - * / // % **`
  - `+= -= *= /= _`
  - operator precedence
  - math module
  - built-in functions

Program structure and functional characteristics

```
import matplotlib.pyplot as plt
import math

n = int(input())    # data points
if n < 10:
    n = 10
x = []
y = []
for k in range(n):
    x.append(k*16*math.pi/n)
    y.append(0.1*k*math.sin(x[k]))
plt.plot(x, y)
plt.show()
```

Program structure and functional characteristics

Execute each instructions line-by-line from top to bottom

```
import matplotlib.pyplot as plt
import math

n = int(input())    # data points
if n < 10:
    n = 10
x = []
y = []
for k in range(n):
    x.append(k*16*math.pi/n)
    y.append(0.1*k*math.sin(x[k]))
plt.plot(x, y)
plt.show()
```

Program structure and functional characteristics

Execute each instructions line-by-line from top to bottom

Some instruction is selection one

```
import matplotlib.pyplot as plt
import math

n = int(input())    # data points
if n < 10:
    n = 10
x = []
y = []
for k in range(n):
    x.append(k*16*math.pi/n)
    y.append(0.1*k*math.sin(x[k]))
plt.plot(x, y)
plt.show()
```

Program structure and functional characteristics

Execute each instructions line-by-line from top to bottom

Some instruction is selection

Some instruction is iteration

```
import matplotlib.pyplot as plt
import math

n = int(input())    # data points
if n < 10:
    n = 10
x = []
y = []
for k in range(n):
    x.append(k*16*math.pi/n)
    y.append(0.1*k*math.sin(x[k]))
plt.plot(x, y)
plt.show()
```

Program structure and functional characteristics

100th Anniversary of Chula Engineering 2013

Execute each instructions line-by-line from top to bottom

Some instruction is selection

Some instruction is iteration

Commentts

```
import matplotlib.pyplot as plt
import math

n = int(input())    # data points
if n < 10:
    n = 10
x = []
y = []
for k in range(n):
    x.append(k*16*math.pi/n)
    y.append(0.1*k*math.sin(x[k]))
plt.plot(x, y)
plt.show()
```

python powered

Program structure and functional characteristics

Execute each instructions line-by-line from top to bottom

Some instruction is selection

Some instruction is iteration

```
import matplotlib.pyplot as plt
import math

n = int(input())    # data points
if n < 10:
    n = 10
x = []
y = []
for k in range(n):
    x.append(k*16*math.pi/n)
    y.append(0.1*k*math.sin(x[k]))
plt.plot(x, y)
plt.show()
```

Program structure and functional characteristics

Execute each instructions line-by-line from top to bottom

Some instruction is selection

Some instruction is iteration

```
import matplotlib.pyplot as plt
import math

n = int(input())    # data points
if n < 10:
    n = 10
x = []
y = []
for k in range(n):
    x.append(k*16*math.pi/n)
    y.append(0.1*k*math.sin(x[k]))
plt.plot(x, y)
plt.show()
```

Basic data types: String and Number

- String
  - "Hello" 'Hello'
  - "Hello Pyton" 'Hello Python'
  - "12345" '12345'
  - "" ''

Basic data types: String and Number

- String
  - "Hello" 'Hello'
  - "Hello Pyton" 'Hello Python'
  - "12345" '12345'
  - "" ''
- Number
  - 1234, 0 (integer)
  - -1234.0, 1.5E-15 ($1.5\ x\ 10^{-15}$) (float)

Operator: plus

- Concatenation for strings
  - `"Hello"+" "+"!"` (got `"Hello !"`)
  - `"1"+"1"` (got `"11"`)

Operator: plus

- Concatenation for strings
  - `"Hello"+" "+"!"` (got `"Hello !"`)
  - `"1"+"1"` (got `"11"`)
- Addition for numbers
  - `1 + 1` (got 2)
  - `1 + 1.0` (got `2.0`)

Operator: plus

- Concatenation for strings
  - `"Hello"+" "+"!"` (got `"Hello !"`)
  - `"1"+"1"` (got `"11"`)
- Addition for numbers
  - `1 + 1` (got 2)
  - `1 + 1.0` (got `2.0`)
- Bad example (run-time error)
  - `"1" + 1`

print(…)

- Program

```
print("Hello")
print("Python")
print("Hello"  + " " + "Python")
print("a", "b", "c", 1, 2, 3)
print("1+1 =", (1+1))
```

print(…)

- Program

```
print("Hello")
print("Python")
print("Hello"  + " " + "Python")
print("a", "b", "c", 1, 2, 3)
print("1+1 =", (1+1))
```

- Output

```
Hello
Python
Hello Python
a b c 1 2 3
1+1 = 2
```

Variables

- variables
  - names
  - untyped, changeable in run-time

Variables

- variables
  - names
  - untyped, changeable in run-time

```
name = "Python"
lang = name
```

store `"Python"` in `name`

store values in `name` in `lang`

making their values same

Variables

- variables
  - names
  - untyped, changeable in run-time

```
name = "Python"
lang = name
```

store "Python" in name
store values in name in lang
making their values same

```
a = 1          # a = 1
b = 2          # a = 1; b = 2
c = a          # a = 1; b = 2; c = 1
d = c + b      # a = 1; b = 2; c = 1; d = 3
d = d + 5      # a = 1; b = 2; c = 2; d = 8
```

Variable name

- Consists of characters, numbers and _
- Case-sensitive
- Cannot start with a number

Variable name

- Consists of characters, numbers and _
- Case-sensitive
- Cannot start with a number

```
and        as         assert    break     class
continue   def        del       elif      else
except     exec       finally   for       from
global     if         import    in        is
lambda     nonlocal   not       or        pass
raise      return     try       while     with
yield      True       False     None
```

Basic data types

- `str` texts
- `int` integers
- `float` floating-point numbers
- `bool` Boolean
- `list` ordered set (aka arrays)

  [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55]

  `["MO", "TU", "WE", "TH", "FR"]`

  `[6230012021, [2110101, 2301107]]`

  `[]`

Examples: Variables and data types

```
first_name = "Ranee"                    # str
last_name  = "Campen"                   # str
aka        = "Bella"                        # str
age        = 29                             # int
height     = 1.65                       # float
is_single  = True                       # bool
birth_date = [24, 12, 1989]             # list
tv_series  = ["Roy Marn", "Plearn Boon",
              "Bubphe Sanniwat", "Krong Kam"]
```

Examples: Variables and data types

```
first_name = "Ranee"                    # str
last_name  = "Campen"                   # str
aka        = "Bella"                        # str
age        = 29                             # int
height     = 1.65                       # float
is_single  = True                       # bool
birth_date = [24, 12, 1989]             # list
tv_series  = ["Roy Marn", "Plearn Boon",
                 "Bubphe Sanniwat", "Krong Kam"]
```

```
print("Name:", first_name, last_name)
print("Age:", age)
print("Aka:", aka, "or", aka[0:4:1])
d = birth_date[0]
m = birth_date[1]
y = birth_date[2]
print("Born:", str(d)+"/"+str(m)+"/"+str(y))
```

Wil explain management of different data types later

Type conversions

- Most values can be typed as string using `str`
- Some values can be typed to `int` or `float`

```
s1 = "123"
s2 = " 456"
n = int(s1) + int(s2) # 579
f = float(s1) + float(s2) # 579.0
print(s1+s2, n, f)
print(s1+s2 + ", " + str(n) + "," + str(f))
```

Type conversions

- Most values can be typed as string using `str`
- Some values can be typed to `int` or `float`

```
s1 = "123"
s2 = " 456"
n = int(s1) + int(s2) # 579
f = float(s1) + float(s2) # 579.0
print(s1+s2, n, f)
print(s1+s2 + ", " + str(n) + "," + str(f))
```

```
a = int(1) # a = 1
b = int(1.9) # b = 1
c = int(1.9 + 0.5) # c = 1
d = float(1.1) # d = 1.1
e = float(1) # e = 1.0
f = str("string") # f = "string"
```

input() accepts string from keyboard

input() accepts key-in value as string and the statement assigns it to name

Program:

```
name = input()
```

input() accepts string from keyboard

input() accepts key-in value as string and the statement assigns it to name

Program:

```
name = input()
```

After keying in Python we get

Output:

```
Python
```

input() accepts string from keyboard

input() accepts key-in value as string and the statement assigns it to name

Program:

```
name = input()
```

After keying in Python we get

Output:

```
Python
Python is a very easy
We use Python in our class.
HelloPython.
```

Accepting numeric input

Program:

```
x = input()
d = float(x)
perimeter = d + d + d + d
print("Perimeter of square =", perimeter)
```

After keying in 12 we get

Accepting numeric input

Program:

```
x = input()
d = float(x)
perimeter = d + d + d + d
print("Perimeter of square =", perimeter)
```

After keying in 12 we get

Output:

```
Perimeter of square = 48.0
```

Accepting numeric input

Program:

```
x = input()
d = float(x)
perimeter = d + d + d + d
print("Perimeter of square =", perimeter)
```

After keying in 12 we get

Output:

```
Perimeter of square = 48.0
```

```
x = input()
d = float(x)
...
```

→

```
d = float(input())
...
```

รับจำนวนจริง

```
x = input()
d = int(x)
...
```

→

```
d = int(input())
...
```

รับจำนวนเต็ม

Basic arithmetic operations

+	Addition
-	Subtraction
*	Multiplication
/	Division
//	Rounded division
%	Remainder
**	Exponentiation
=	Assignment

Basic arithmetic operations

+	Addition
-	Subtraction
*	Multiplication
/	Division
//	Rounded division
%	Remainder
**	Exponentiation
=	Assignment

```
a = 5 + 2          # 7
b = 5 - 2          # 3
c = 5 * 2          # 10
d = 5 ** 2         # 25
e = 5 / 2          # 2.5
f = 5 // 2         # 2
g = 5 % 2          # 1

a,b,c = c,a,b      # a = 10
                   # b = 7
                   # c = 3
a,b = b,a          # Exchange a and b
```

Tips

```
a = float(input())
a = -a                          # change the sign of number
b = int(a)                      # round a down
c = int(a+0.5)                  # round a up or down
d = a – int(a)                  # extract decimal point of a
d = a % 1                       # same effect as previously
e = b % 10                      # least significant digit of b
f = b // 10 % 10                # second to least sig. digit of b
g = b // 10**4 % 10             # fifth digit of b

h = input()                     # h stores string value
h = int(h)                      # h stores integer value
```

Exercise: Celsius input, display Fahrenheit

Input		Output
0	→	32
100	→	312

```
celsius = float(input())
```

Augmented assignments


```
a = 10
a = a + 5          # 15
a = a - 1          # 14
a = a * 2          # 28
a = a % 10         # 8
a = a ** 2         # 64
a = a // 10        # 6
```

```
a = 10
a += 5             # 15
a -= 1             # 14
a *= 2             # 28
a %= 10            # 8
a **= 2            # 64
a //= 10           # 6
```

math module

```
import math

degree = float(input())
radian = degree * 3.14159 / 180
radian = degree * math.pi /180
radian = math.radians(degree)
s = math.sin(radian)
c = math.cos(radian)
g = math.log( 1E100, 10 ) # g = 100.0
```

math module

```
import math

degree = float(input())
radian = degree * 3.14159 / 180
radian = degree * math.pi /180
radian = math.radians(degree)
s = math.sin(radian)
c = math.cos(radian)
g = math.log( 1E100, 10 ) # g = 100.0
```

A lot of mathematical built-in functions in math module

Bult-in functions

```
a = abs(-2)                 # a = 2
b = round(2/3, 2)           # b = 0.67
c = max([4, 1, 5, 3])       # c = 5
d = min([4, 1, 5, 3])       # d = 1
e = sum([4, 1, 5, 3])       # e = 13
f = len([4, 1, 5, 3])       # f = 4
g = str(1234)               # g = "1234"
h = int("123")              # h = 123
i = float("-123.4")         # j = -123.4
j = input()
print(a, b, c, d, e, f, g, h, i, j)
```

Bult-in functions


```
a = abs(-2)                 # a = 2
b = round(2/3, 2)           # b = 0.67
c = max([4, 1, 5, 3])       # c = 5
d = min([4, 1, 5, 3])       # d = 1
e = sum([4, 1, 5, 3])       # e = 13
f = len([4, 1, 5, 3])       # f = 4
g = str(1234)               # g = "1234"
h = int("123")              # h = 123
i = float("-123.4")         # j = -123.4
j = input()
print(a, b, c, d, e, f, g, h, i, j)
```

You will learn how to define your own functions in subsequent chapters

Nesting different functions


```
x1 = input()
x2 = int(x1)
x3 = abs(x2)
x4 = 1 + x3
x5 = str(x4)
print("x5 = " + x5)
```

```
print("x5 = " + str(1+abs(int(input()))))
```

Too much nesting will make your codes confusing

Operator precedence: 2+3*4 = ?

A code statement with many operators will be calculated based on the following order

- (…)
- **
- - (unary)
- * / // %
- + -
- In clause with many operators with same precedence, do them left to right (except for ** with right to left)

```
2 * 3 + 8 / -(2 - 4) - 2**2**3
2 * 3 + 8 / -(-2) - 2**2**3
2 * 3 + 8 / -(-2) - 2**8
2 * 3 + 8 / -(-2) - 256
2 * 3 + 8 / 2 - 256
6 + 8 / 2 - 256
6 + 4.0 - 256
10.0 - 256
```

Example: area, circumference calculation

100th Anniversary of Chula Engineering 2013

Program:

```
radius = float(input())
area = math.pi * radius**2
circum = 2 *math.pi * radius
print("Area =", round(area, 2))
print("Circumference =", round(circum, 2))
```

After keying in `4.5` will get

Example: area, circumference calculation

100th Anniversary of Chula Engineering 2013

Program:

```
radius = float(input())
area = math.pi * radius**2
circum = 2 *math.pi * radius
print("Area =", round(area, 2))
print("Circumference =", round(circum, 2))
```

After keying in `4.5` will get

Output:

```
Area = 63.62
Circumference = 28.27
```

python powered

Example: student id

6 2 3 0 0 1 2 0 2 1

Example: student id

6 2 3 0 0 1 2 0 2 1

Example: student id

6 2 3 0 0 1 2 0 2 1

Admission year

0 = Doctor of Philosophy
1 = Master's degree
3 = Bachelor of Science
4 = Bachelor of Humanities
7 = Master of Science
8 = Master of Humanities

21 Faculty of Engineering
22 Faculty of Arts
23 Faculty of Science
24 Faculty of Political Science
…

Example: student id to (faculty, year, degree)

Program:

```
stu_id = int(input())
print("Student ID:", stu_id)
fac_code = stu_id % 100        # Two least significant digits
year_in = 2500 + stu_id//10**8 # From two most significant digits
deg_code = stu_id//10**7 % 10
print("Faculty code:", fac_code)
print("Enrollment year:", year_in)
print("Academic degree:", deg_code)
```

Example: student id to (faculty, year, degree)

Program:

```
stu_id = int(input())
print("Student ID:", stu_id)
fac_code = stu_id % 100        # Two least significant digits
year_in = 2500 + stu_id//10**8 # From two most significant digits
deg_code = stu_id//10**7 % 10
print("Faculty code:", fac_code)
print("Enrollment year:", year_in)
print("Academic degree:", deg_code)
```

After keying in 6230012021 will get

Output:

```
Student ID: 6230012021
Faculty code: 21
Enrollment year: 2562
Academic degree: 3
```

Show each digit of student id


```
sid = int(input())            # sid, say, = 6231020121
d = sid % 10; print(d); sid //= 10 # 621102012
d = sid % 10; print(d); sid //= 10 # 62110201
d = sid % 10; print(d); sid //= 10 # 6211020
d = sid % 10; print(d); sid //= 10 # 621102
d = sid % 10; print(d); sid //= 10 # 62110
d = sid % 10; print(d); sid //= 10 # 6211
d = sid % 10; print(d); sid //= 10 # 621
d = sid % 10; print(d); sid //= 10 # 62
d = sid % 10; print(d); sid //= 10 # 6
d = sid % 10; print(d);
```

punctuation ; separate different statements in same line

Exercise: Stirling formula

$$n! \sim \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$$

```
import math
n = int(input())
```

Exercise: roots of $ax^2 + bx + c = 0$

Accept key-in a, b, c
Calculate and display two roots

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$